Agile Software Development.com

10 contrats pour votre prochain Projet Logiciel Agile

Auteur : Peter Stevens

Son article du 29 avril 2009 : http://agilesoftwaredevelopment.com/blog/peterstev/10-agile-contracts
Son blog : http://www.scrum-breakfast.com/

En tant que client ou fournisseur de prestations logicielles au début d'un projet de développement d'un logiciel, vous savez qu'il y a beaucoup trop en jeu pour travailler sur la base d'un simple accord verbal. Un contrat est en fait juste un ensemble de règles de jeu écrites. Les bonnes règles augmentent les chances de succès pour les deux parties. Les mauvaises règles rendent la coopération difficile et empêchent le progrès. Quelles sont les règles de jeu disponibles et quelle est la meilleure approche sur un projet agile ?

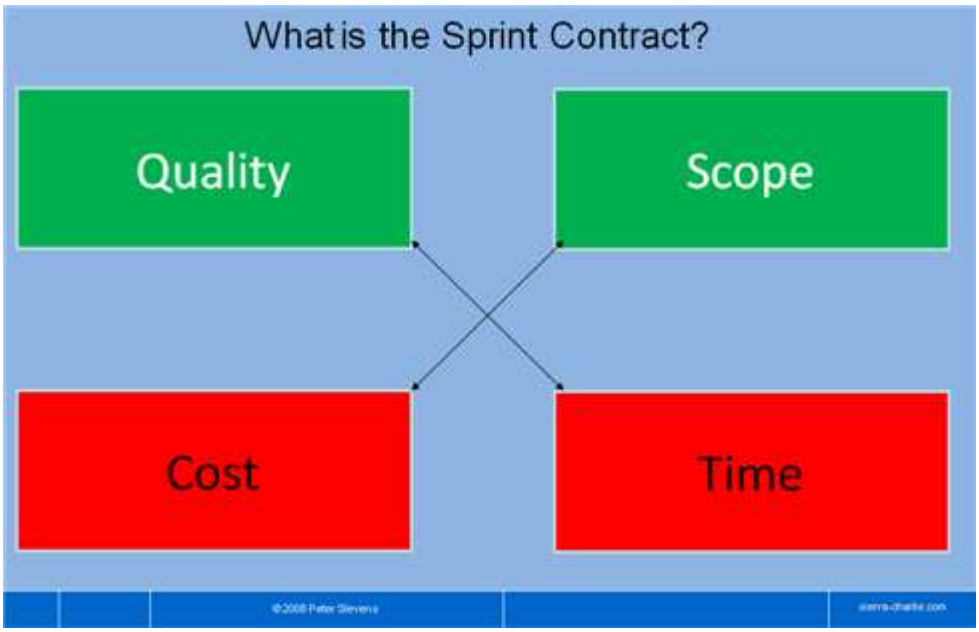
La semaine dernière, nous nous sommes penchés sur l'objectif et le contenu d'un contrat et nous avons identifié des critères d'évaluation des contrats pour des projets Scrum et agiles. J'ai proposé 4 points de comparaison entre les différents types de contrats :

- Comment est structuré le contrat ?
- Comment traite-t-il les changements de périmètre (exigences) ?
- Comment répartit-il les risques et la rémunération entre le client et le fournisseur ?
- Quel modèle de relation client privilégie-t-il : concurrentiel (ma victoire est votre perte), coopératif (gagnant/gagnant), indifférent (je m'en moque-vous perdez) ou statistique (pile, je gagne – face, vous perdez) ?

Cette semaine, prenons quelques formes possibles de contrats, et observons comment cela s'applique sur des projets de développement agile et Scrum :

1. le contrat au sprint
2. le forfait / périmètre figé
3. l'assistance technique
4. l'assistance technique avec un périmètre figé et un budget limité
5. l'assistance technique avec un périmètre variable et un budget limité
6. le développement par phase
7. les clauses de bonus / pénalités
8. le bénéfice fixé à l'avance
9. le profit pour rien, les changements à discrétion
10. le projet commun

1. Le contrat au sprint



Travaillant avec Scrum, j'ai trouvé la métaphore du « contrat au sprint » très utile pour comprendre (et parfois renforcer) les relations entre le Product Owner et l'Equipe de développement.

Structure : ce n'est pas réellement un contrat commercial, mais simplement un accord entre le Product Owner et l'Equipe sur un seul sprint.

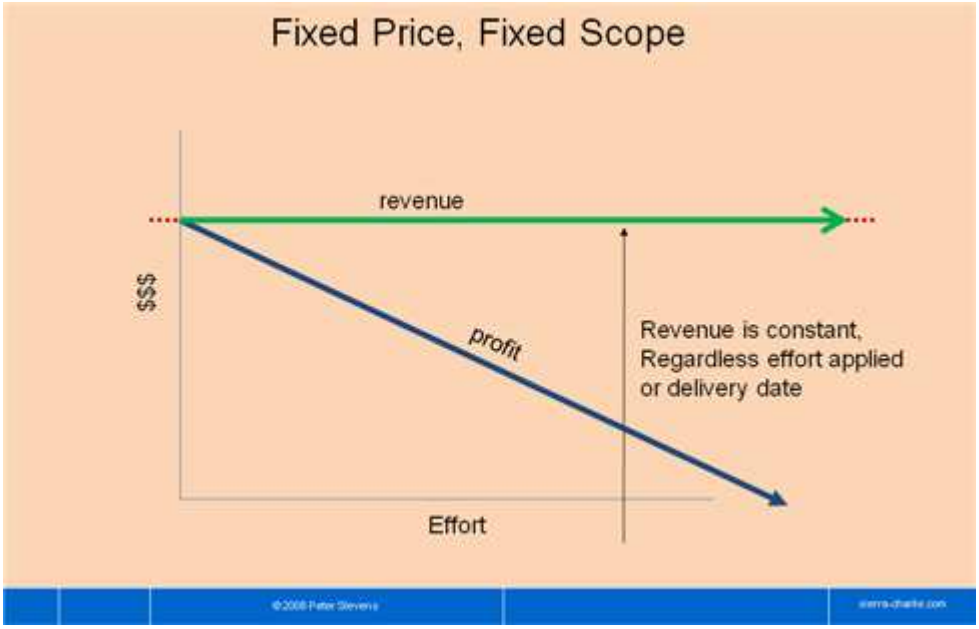
Périmètre : l'Equipe de développement est d'accord pour faire de son mieux et livrer à la fin du sprint un ensemble validé de fonctionnalités (périmètre) avec une qualité donnée (idéalement elle livre ce qu'elle a promis, voire un peu plus). Le Product Owner est d'accord pour ne rien changer pendant toute la durée du sprint.

Risque : un projet Scrum peut être considéré comme une suite de mini projets dont les paramètres sont fixés : Délai (longueur du sprint), Périmètre (backlog du sprint), Qualité (définition de « Terminé ») et les Charges (taille de l'équipe $\times$ longueur du sprint). Seul le périmètre peut changer à chaque sprint.

Conseil : confirmer le « contrat au sprint » par E-mail ou le préciser sur le wiki du projet à chaque début de sprint est généralement une bonne idée. Cela renforce la confiance, quelle que soit la forme contractuelle sous-jacente.

Conseil : le contrat au sprint peut être référencé dans le contrat commercial. J'ai finalement constaté que le contrat commercial pouvait se réduire à un accord d'assistance technique d'une page, avec éventuellement un plafonnement des budgets sur la prochaine grosse livraison.

2. Le forfait / périmètre figé



Structure : mettez-vous d'accord sur les livrables, produisez-les. Envoyez une facture. Les clients aiment les projets au forfait parce que cela les sécurise (ou du moins le pensent-ils).

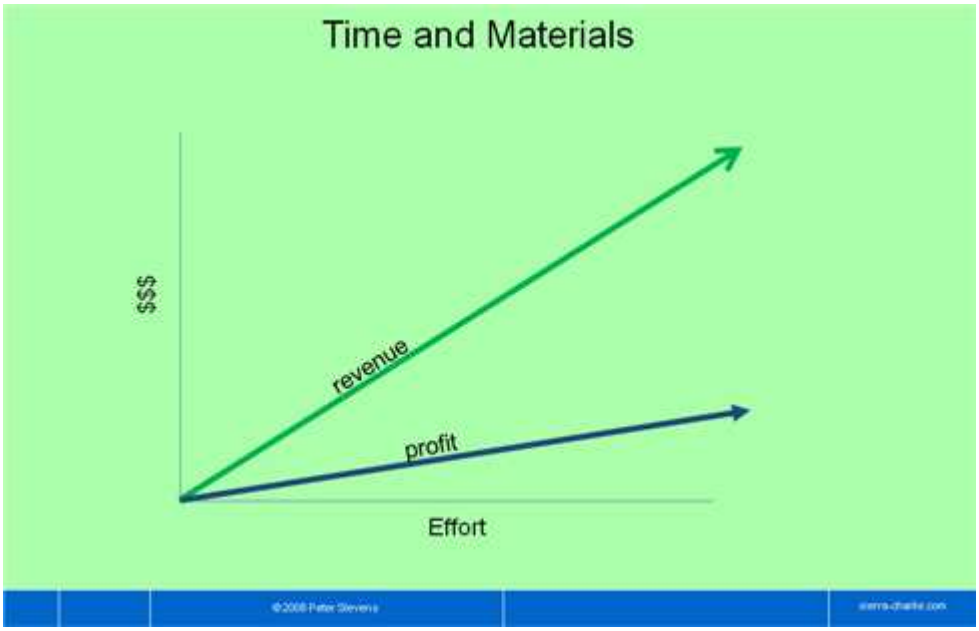
Changements de périmètre : cette expression dit tout, n'est-ce pas ? Le jeu de la demande de changement (correction : processus de demande de changement) est destiné à limiter la portée des changements. Ce processus est coûteux, et les changements ne sont généralement pas prévisibles. Puisque par définition le client souhaite étendre le périmètre, terminer le projet devient difficile. Comme le fournisseur souhaite que le client soit satisfait, il produit généralement. Le mot « et cetera" est très dangereux dans les spécifications d'une exigence au forfait.

Risque : un risque évident est porté par le fournisseur : si son estimation est mauvaise, le projet perdra de l'argent. Les risques moins évidents sont le jeu de la demande de changement grâce auquel le fournisseur négocie des avenants pour les changements de périmètre. Si le fournisseur a lourdement sous-estimé l'effort ou le risque, ou estimé un coût anormalement bas, les pertes peuvent même menacer l'existence du fournisseur, nouveau problème pour le client.

Relations : de concurrentiel à indifférent. Le client souhaite généralement avoir plus et le fournisseur souhaite en faire moins. Le fournisseur souhaite que le client soit heureux, donc généralement il produit.

Conseil : spécifiez les exigences fonctionnelles avec des user stories. Je présenterai les stratégies pour atteindre l'objectif sur un projet au forfait dans un prochain article.

3. L’assistance technique



Structure : travaillez pendant 1 mois, puis envoyez une facture au client. Les fournisseurs aiment ça puisque c’est le client qui porte le risque en changeant d’avis sur le périmètre.

Périmètre : non fixé a priori. Tôt ou tard, le client ne veut pas payer plus et le projet arrive à son terme par la force des choses.

Risques : portés à 100% par le client. Le fournisseur a quand même un peu intérêt à garder des prix bas. Il doit également veiller à facturer des charges et des frais légitimes.

Relations : indifférent. Le fournisseur est heureux lorsqu’il y a plus de travail parce que qui dit plus de travail dit plus d’argent.

Conseil : recommandé pour les projets où le client peut mieux gérer le risque que le fournisseur. Ceci est souvent combiné avec un plafonnement des budgets. Cela marchera d’autant mieux si le périmètre est correctement maîtrisé.

4. L'assistance technique avec un périmètre figé et un budget limité

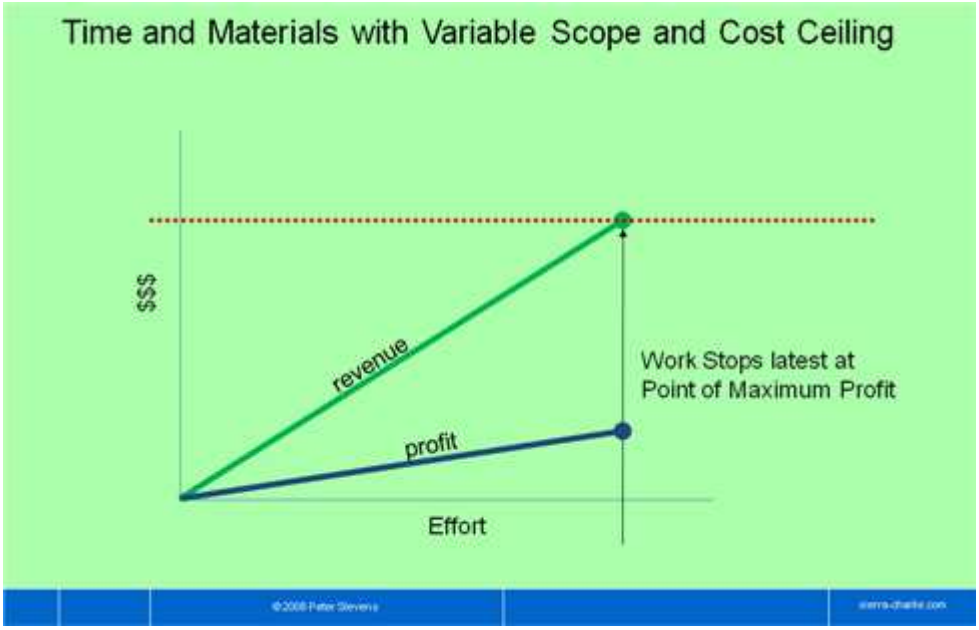
Structure : identique au "forfait / périmètre figé", sauf si le fournisseur finit plus tôt, le projet coûte moins cher car seul les charges consommées sont facturées.

Périmètre : identique au "forfait / périmètre figé".

Risques : il s'agit du "meilleur des deux mondes" pour le client. Si cela réclame moins de charges que prévu, cela coûte moins cher. Mais une fois que le plafond est atteint, on repasse en mode forfait.

Relation : statistique. Du point de vue fournisseur, l'objectif est d'atteindre exactement le plafond. Il n'y a pas d'intérêt pour lui de livrer avant que le budget maximum ait été consommé. De son côté, le client a sans doute appréhendé en interne le projet comme un projet au forfait et il n'y a pas d'intérêt pour lui de réduire le périmètre, même légèrement, pour économiser du budget.

5. L'assistance technique avec un périmètre variable et un budget limité



Structure : identique à "l'assistance technique", sauf que le plafond qui limite le risque financier pour le client.

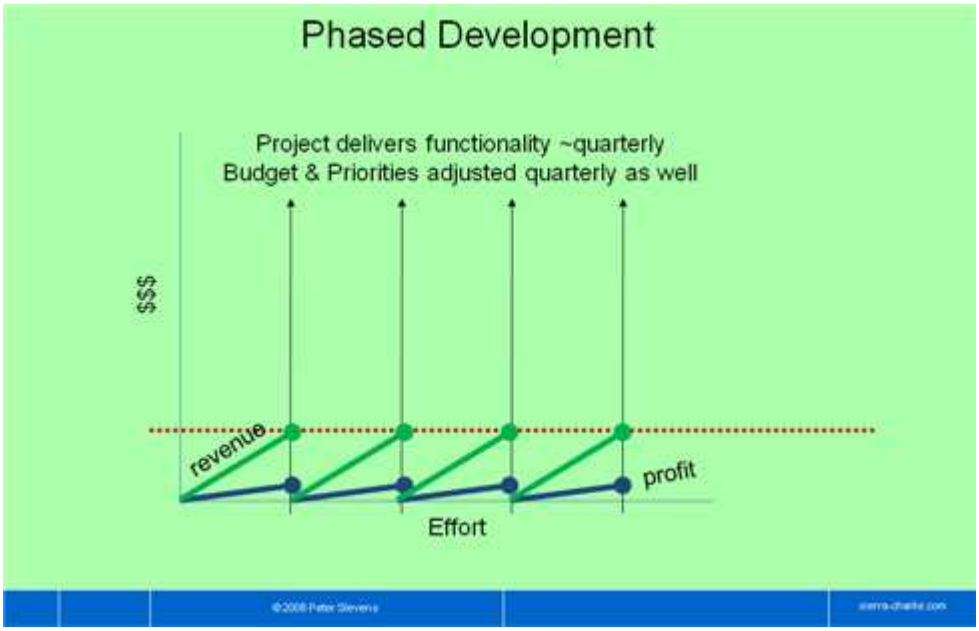
Périmètre : identique au "forfait / périmètre figé".

Risques : le budget expirera avant d'avoir atteint la valeur métier souhaité par le client. Le client n'aura pas tout ce qu'il avait demandé.

Relations : coopératif. La combinaison « budget limité » et « périmètre variable » motive le client et le fournisseur pour atteindre la valeur métier souhaitée avec le budget disponible.

Conseil : d'expérience, je voudrais dire que cette forme de contrat s'applique bien à Scrum. Le client garde le contrôle sur chaque sprint. Une relation coopérative, constructive, signifie que même si des problèmes apparaissent, le client et le fournisseur sont prêts à trouver une solution mutuellement acceptable.

6. Le développement par phase



Structure : budget pour une version trimestrielle et validation d'un budget supplémentaire après chaque livraison réussie.

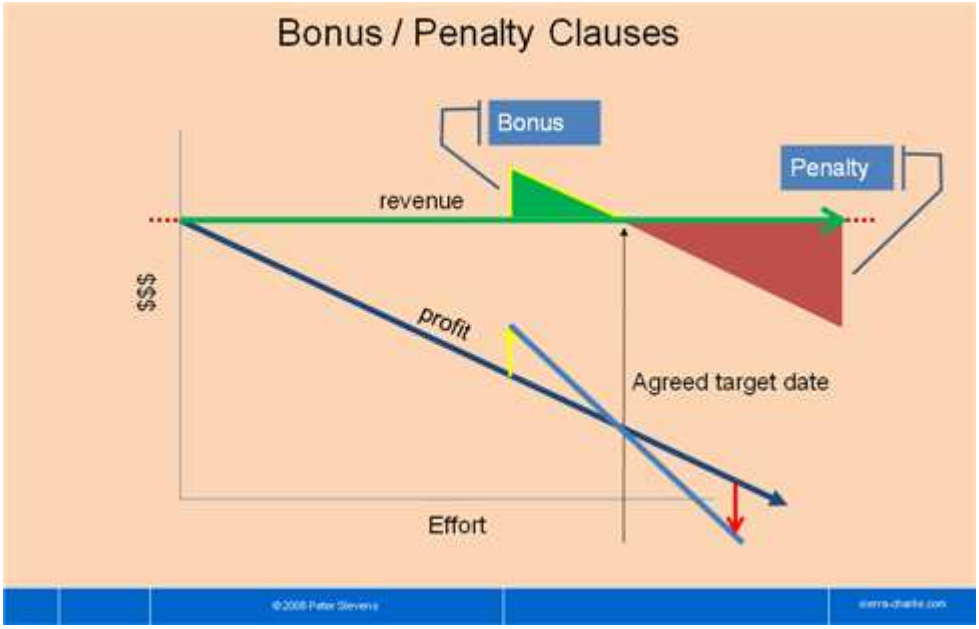
Changements de périmètre : non défini explicitement par le modèle. Les versions sont livrées à intervalles réguliers (boîtes de temps). Le fait de savoir qu'une autre version sera réalisée le trimestre suivant facilite la décision de différer la mise en œuvre d'une fonctionnalité pour respecter le rythme trimestriel.

Risques : le risque client est limité à un coût trimestriel de développements.

Relations : coopératif. Le client et le fournisseur ont intérêts à ce que chaque livraison se passe bien, de telle façon que la réalisation de la version suivante sera bien budgétée.

Conseil : les sociétés de capital-risque opèrent souvent de cette manière. Cela se combine très bien avec « l'assistance technique avec un périmètre variable et un budget limité ». J'ai vraiment apprécié travailler selon ce modèle. Nous spécifions simplement le but de la version, le taux horaire et le budget à ne pas dépasser dans le contrat commercial. Les autres clauses sont abordées dans les contrats des sprints.

7. Les clauses de bonus / pénalités



Structure : le fournisseur reçoit un bonus si le projet est terminé en avance et paye une pénalité s'il est en retard. Le montant du bonus ou de la pénalité est fonction du délai.

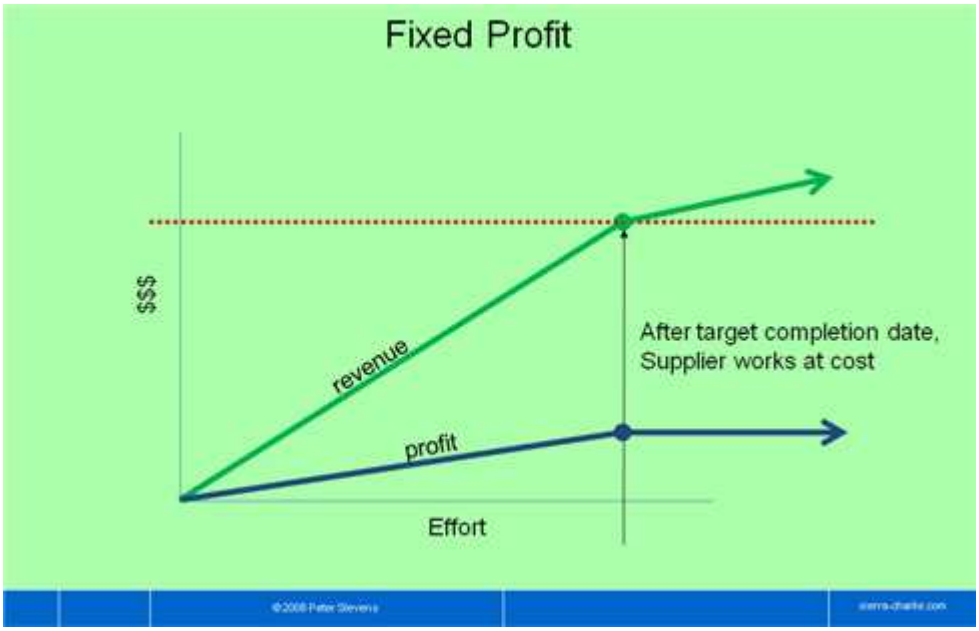
Changements de périmètre : difficile à accepter car les changements de périmètre impacte potentiellement la date de livraison, ce qui sera très difficilement autorisé.

Risques : le client a-t-il également intérêt à terminer plus tôt ? Les arguments concernant le ROI doivent être convaincants et transparents. Sinon, le client obtient une solution meilleur marché en prenant le temps nécessaire.

Relations : peut être coopératif, mais pourrait dégénérer si le client n'a pas vraiment besoin du logiciel à la date convenue.

Conseil : souvent employé dans les projets de construction (routes, tunnels, autoroutes …) pour lesquels cela fonctionne très bien. Les changements de périmètre ne se posent pas et l'espérance de gains significatifs conduit le client à vouloir respecter l'échéance.

8. La bénéfice fixé à l'avance



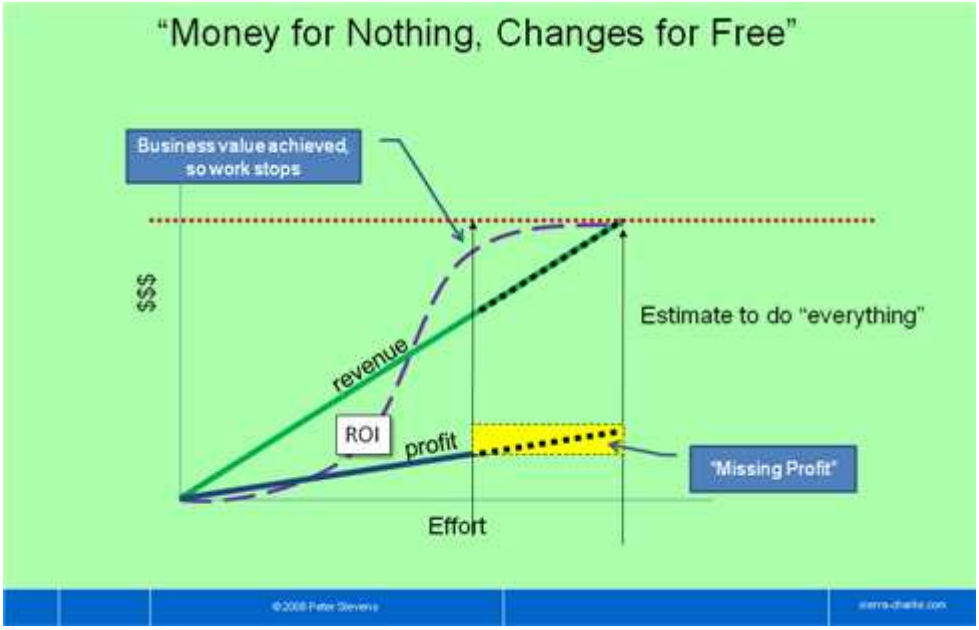
Structure : le budget d'un projet comprend les coûts effectifs et le bénéfice. Les deux parties se mettent préalablement d'accord sur le bénéfice, par exemple 100 000 €. Indépendamment de la fin du projet, le fournisseur est payé des coûts effectifs et du bénéfice fixé à l'avance.

Périmètre : figé.

Risques : partagés. Si le projet se termine plus tôt, le client paye moins, mais le fournisseur garde son bénéfice. Si le projet dépasse le budget initial, le client paye plus, mais le fournisseur ne fait pas plus de bénéfice. Après la date de livraison cible, le fournisseur facture et couvre uniquement ses coûts mais sans bénéfice supplémentaire.

Relations : coopératif. Client et fournisseur ont, tous les deux, intérêt à terminer à l'heure. Le client économise de l'argent et le fournisseur augment sa marge.

9. Le profit pour rien, les changements à discrétion



Structure : cela fonctionne pour des projets logiciels agiles parce qu'il n'y a pas ou très peu de tâches en cours. Après chaque sprint, la fonctionnalité est soit terminée soit non commencée. Le mode de travail est basé sur de l'assistance technique avec un objectif de coût, souvent avec l'intention que le projet ne devrait pas utiliser la totalité du budget. Une fois qu'un certain nombre de fonctionnalités ont été livrées, le client peut constater que suffisamment de valeur métier a été réalisée, que plus aucun développement n'est nécessaire et qu'il peut par conséquent annuler le projet. Des frais d'annulation équivalents au bénéfice théorique doivent être payés.

Périmètre : peut varier. Planifié mais des fonctionnalités non encore implémentées peuvent être remplacées par des stories de la même taille. Chaque fonctionnalité supplémentaire doit être budgétée.

Risques : partagés. Client et fournisseur ont intérêt à terminer le projet plus tôt. Le client paye moins et le fournisseur augmente ses bénéfices.

Conseil : si le budget est dépassé, les règles du « bénéfice fixé par avance » ou du « budget limité » peuvent s'appliquer. L'approche « bénéfice fixé par avance » favorise davantage la relation de coopération.

10. Le projet commun

Structure : deux partenaires investissent sur un produit d'intérêt commun Dans ce cas, il n'est pas question d'argent entre partenaires dabs la phase de développement, mais chacun doit bénéficier d'un ROI, provenant d'un partage des revenus ou des bénéfices suite à l'utilisation du logiciel.

Périmètre : compatible avec les besoins du partenariat.

Risques : la chaîne de décisions peut être longue. Des rivalités peuvent apparaître entre les équipes. Différents modèles d'extraction de la valeur produit peuvent créer des différences de priorité et amoindrir la volonté d'investir.

Conseil : appréhender le projet comme une entreprise séparée. Une équipe en colocation avec un rôle de développement et un rôle de Product Owner. Pensez cohésion et non separation.

Recommandations

Pendant des années, j'ai très bien travaillé avec un contrat du type « développement par phase ». Le contrat initial était un contrat avec un périmètre figé et un budget limité, mais comme nous avons travaillé ensemble et établi le niveau de confiance nécessaire, la notion de contrat a disparu. La confiance, le contrat du sprint et une signature du budget chaque trimestre par la Direction ont parfaitement suffi.

Le mode de contractualisation « le profit pour rien, les changements à discrétion » rend compétitif les processus de développement agile et Scrum.

- En priorisant et livrant itérativement la valeur métier, les chances d'un échec sont considérablement réduites. Cet avantage est partagé avec le client.
- En outre, il s'agit d'un modèle coopératif, il motive le client et le fournisseur pour maintenir des coûts bas.
- La clause de résiliation anticipée récompense la productivité élevée obtenue avec les équipes Scrum. En revanche, cette clause a un peu des airs de « parachute doré », ce qui n'est peut être pas politiquement acceptable dans le climat économique actuel.

Ce type de contrat pourrait fonctionner avec un budget limité mais j'hésiterais si je devais prendre le risque de détériorer ou perdre la relation de coopération.

Les clauses du contrat établissent les bases importantes de la réussite d'un projet. Et le Manifeste Agile a raison : travailler avec un client est plus important que le contrat. Donc, quoi que vous fassiez, gardez une relation positive avec votre client !